Lab Session 03

Control Structures, Decision Making using simple If, If-Else statement.

Objectives:

1. To understand Control Structures,
2. Demonstration of decision Making using simple If, If-Else statement.
3. Design and Compilation of C++ programs based on decision making.

Algorithms

Any solvable computing problem can be solved by executing of a series of actions in a specific order. A procedure for solving a problem in terms of 1. the actions to execute and 2. the order in which the actions execute is called an algorithm. The following example demonstrates that correctly specifying the order in which the actions execute is important. Consider the "rise-and-shine algorithm" followed by one junior executive for getting out of bed and going to work:

Control Structures.

A program is usually not limited to a linear sequence of instructions. During its process it may bifurcate, repeat code or take decisions. For that purpose, C++ provides control structures that serve to specify what has to be done to perform our program.

With the introduction of control sequences we are going to have to introduce a new concept: the block of instructions. A block of instructions is a group of instructions separated by semicolons (;) butgrouped in a block delimited by curly bracket signs: { and }.

Most of the control structures that we will see in this section allow a generic statement as a parameter, this refers to either a single instruction or a block of instructions, as we want. If we want the statement to be a single instruction we do not need to enclose it between curly-brackets ({}). If we want the statement to be more than a single instruction we must enclose them between curly brackets ({}) forming a block of instructions.

Conditional structure: if and else

It is used to execute an instruction or block of instructions only if a condition is fulfilled. Its form is:

```
if (condition) statement
```

where *condition* is the expression that is being evaluated. If this condition is **true**, ***statement*** is executed. If it is false, *statement* is ignored (not executed) and the program continues on the next instruction after the conditional structure.

For example, the following code fragment prints out **x is 100** only if the value stored in variable **x** is indeed 100:

```
if (x == 100)
 cout << "x is 100";
```

If we want more than a single instruction to be executed in case that *condition* is **true** we can specify a *block of instructions* using curly brackets { }:

```
if (x == 100)
 {
  cout << "x is ";
  cout << x;
 }
```

We can additionally specify what we want that happens if the condition is not fulfilled by using the keyword *else*. Its form used in conjunction with **if** is:

```
if (condition) statement1 else statement2
```

For example:

```
if (x == 100)
 cout << "x is 100";
else
  cout << "x is not 100";
```

prints out on the screen **x is 100** if indeed x is worth 100, but if it is not -and only if not- it prints out **x is not 100**.

The if + *else* structures can be concatenated with the intention of verifying a range of values. The following example shows its use telling if the present value stored in **x** is positive, negative or none of the previous, that is to say, equal to zero.

```
if (x > 0)
  cout << "x is positive";
else if (x < 0)
  cout << "x is negative";
```

```
else
  cout << "x is 0";
```

Remember that in case we want more than a single instruction to be executed, we must group them in a *block of instructions* by using curly brackets **{ }**.

Decisions

The decisions in a loop always relate to the same question: Should we do this (the loop body) again?
Programs also need to make these one-time decisions. In a program a decision causes a onetime jump to a different part of the program, depending on the value of an expression.
Decisions can be made in C++ in several ways. The most important is with the if...else statement, which chooses between two alternatives. This statement can be used without the else, as a simple if statement. Another decision statement, switch, creates branches for multiple alternative sections of code, depending on the value of a single variable. Finally, the conditional operator is used in specialized situations. We'll examine each of these constructions.

The if Statement

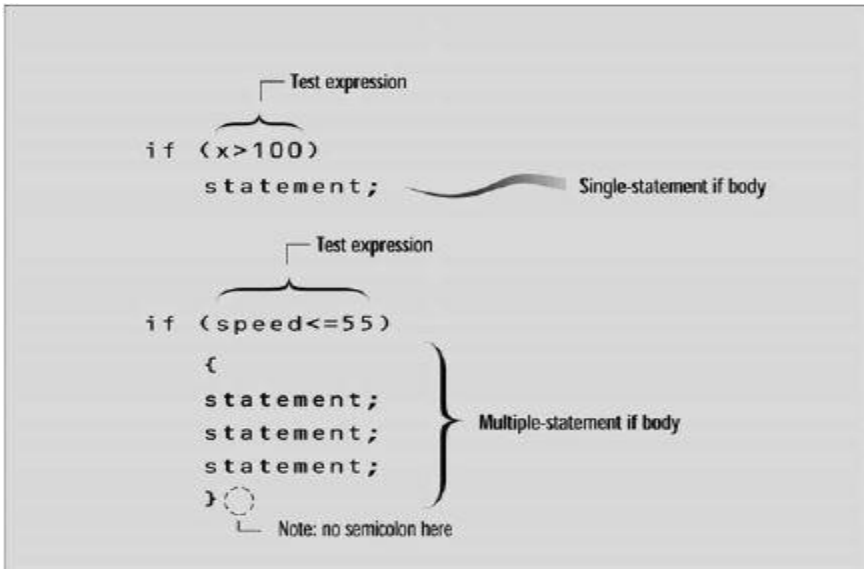
The if statement is the simplest of the decision statements. Our next program, IFDEMO, provides
an example.

```
// ifdemo.cpp
// demonstrates IF statement
#include <iostream>
using namespace std;
int main()
{
int x;
cout << "Enter a number: ";
cin >> x;
if( x > 100 )
cout << "That number is greater than 100\n";
return 0;
}
```

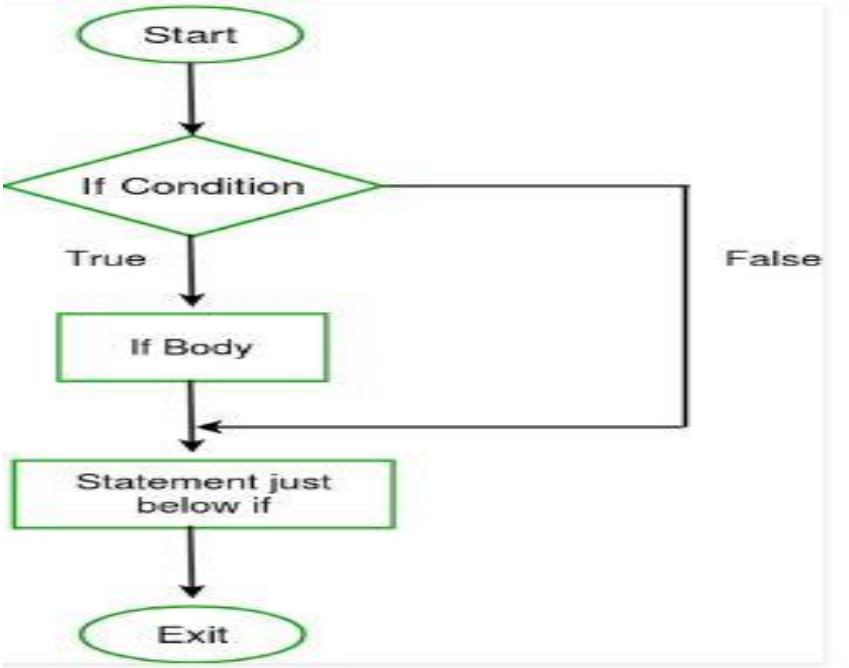
The if keyword is followed by a test expression in parentheses. The syntax of the if statement is shown in Figure 3.7. As you can see, the syntax of if is very much like that of while. The difference is that the statements following the if are executed only once if the test expression is true; the statements following while are executed repeatedly until the test expression becomes false. Figure 3.8 shows the operation of the if statement.

Here's an example of the IFDEMO program's output when the number entered by the user is greater than 100:
Enter a number: 2000
That number is greater than 100
If the number entered is not greater than 100, the program will terminate without printing the second line.

Syntax of if-statement:



Operation of if-statement:



Multiple Statements in the if Body

As in loops, the code in an if body can consist of a single statement—as shown in the IFDEMO example—or a block of statements delimited by braces. This variation on IFDEMO, called IF2, shows how that looks.

```
#include <iostream>
using namespace std;
int main()
{
int x;
cout << “Enter a number: “;
cin >> x;
if( x > 100 )
{
cout << “The number “ << x;
cout << “ is greater than 100\n”;
}
return 0;
}
```

Here’s some output from IF2:

```
Enter a number: 12345
The number 12345 is greater than 100
```

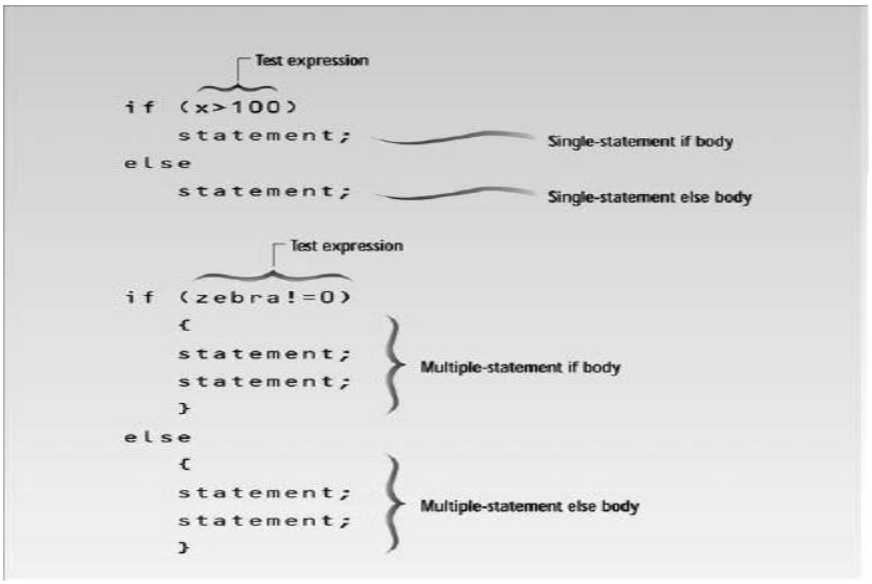
The if...else Statement

The if statement lets you do something if a condition is true. If it isn't true, nothing happens. But suppose we want to do one thing if a condition is true, and do something else if it's false. That's where the if...else statement comes in. It consists of an if statement, followed by a statement or block of statements, followed by the keyword else, followed by *another* statement or block of statements. The syntax is shown in Figure.

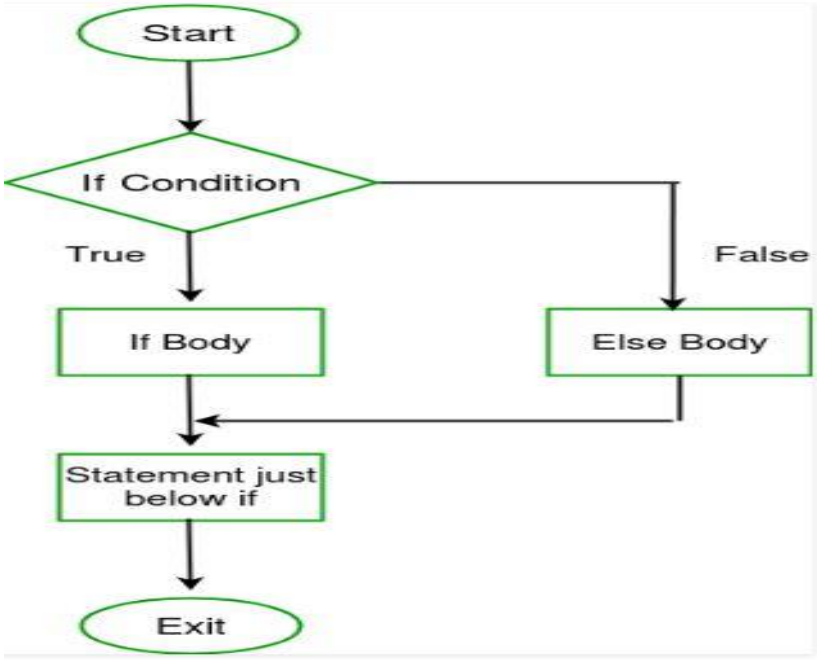
```
#include <iostream>
using namespace std;

int main()
{
int x;
cout << "\nEnter a number: ";
cin >> x;
if( x > 100 )
cout << "That number is greater than 100\n";
else
cout << "That number is not greater than 100\n";
return 0;
}
```

Syntax of if-Else statement:



Operation of if-Else statement:



Assignment:

1. **What is PseudoCode?**
2. **Write a program whether an interger entered is positive or negative number entered by the user.**
3. **Write a program that takes students marks from user, if the obtained marks is less than 60 prints “Fail” otherwise “Pass”.**